Continued Fraction Expansion of Real Roots of Polynomial Systems

Angelos Mantzaflaris, Bernard Mourrain, Elias Tsigaridas
GALAAD, INRIA Sophia Antipolis
[FirstName.LastName]@sophia.inria.fr

ABSTRACT

We present a new algorithm for isolating the real roots of a system of multivariate polynomials, given in the monomial basis. It is inspired by existing subdivision methods in the Bernstein basis; it can be seen as generalization of the univariate continued fraction algorithm or alternatively as a fully analog of Bernstein subdivision in the monomial basis. The representation of the subdivided domains is done through homographies, which allows us to use only integer arithmetic and to treat efficiently unbounded regions. We use univariate bounding functions, projection and preconditionning techniques to reduce the domain of search. The resulting boxes have optimized rational coordinates, corresponding to the first terms of the continued fraction expansion of the real roots. An extension of Vincent's theorem to multivariate polynomials is proved and used for the termination of the algorithm. New complexity bounds are provided for a simplified version of the algorithm. Examples computed with a preliminary C++ implementation illustrate the approach.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*algebraic algorithms*

General Terms

Algorithms, Theory

Keywords

subdivision algorithm, homography, tensor monomial basis, continued fractions, C++ implementation

1. INTRODUCTION

The problem of computing roots of univariate polynomials has a long mathematical history [14]. Recently, some new investigations focused on subdivision methods, where root localization is based on simple tests such as *Descartes' Rule of Signs* and its variant in the Bernstein basis [13, 7, 4]. Complexity analysis was developed for univariate integer polynomial taking into account the bitsize of the coefficients, and providing a good understanding of their behavior from a theoretical and practical point of view. Approximation and bounding techniques have been developed [2] to improve the local speed of convergence to the roots.

Even more recently a new attention has been given to continued fraction algorithms (CF), see e.g. [16, 18] and references therein. They differ from previous subdivision-based algorithms in that instead of bisecting a given initial interval and thus producing a binary expansion of the real roots, they compute continued fraction expansions of these roots. The algorithm relies heavily on computations of lower bounds of the positive real roots, and different ways of computing such bounds lead to different variants of the algorithm. The best known worst-case complexity of CF is $\widetilde{\mathcal{O}}_B(d^5\tau^2)$ [16], while its average complexity is $\widetilde{\mathcal{O}}_B(d^3\tau)$, thus being the only complexity result that matches, even in the average the complexity bounds of numerical algorithms [15]. Moreover, the algorithm seems to be the most efficient in practice [6, 18].

Subdivision methods for the approximation of isolated roots of multivariate systems are also investigated but their analysis is much less advanced. In [17], the authors used tensor product representation in Bernstein basis and domain reduction techniques based on the convex hull property to speed up the convergence and reduce the number of subdivisions. In [5], the emphasis is put on the subdivision process, and stopping criterion based on the normal cone to the surface patch. In [12], this approach has been improved by introducing pre-conditioning and univariate-solver steps. The complexity of the method is also analyzed in terms of intrinsic differential invariants.

This work is in the spirit of [12]. The novelty of our approach is the presentation of a tensor-monomial basis algorithm that generalizes the univariate continued fraction algorithm and does not assume generic position. We apply a subdivision approach also exploiting certain properties of the Bernstein polynomial representation, even though no basis conversion takes place.

Our contributions are as follows. We propose a new adaptive algorithm for polynomial system real solving that acts in monomial basis, and exploits the continued fraction expansion of (the coordinates of) the real roots. This yields the best rational approximation of the real roots. All computations are performed with integers, thus this is a division-free algorithm. We propose a first step towards the generalization of Vincent's theorem to the multivariate case (Th. 4.2) We perform a (bit) complexity analysis of the algorithm, when oracles for lower bounds and counting the real roots are available (Prop. 5.2) and we propose non-trivial improvements for reducing the total complexity even more (Sec. 5.3). In all cases the bounds that we derive for the multivariate case, match the best known ones for the univariate case, if we restrict ourselves to $n = 1$.

1.1 Notation

For a polynomial $f \in \mathbb{R}[x_1, .., x_n]$, $\mathsf{deg}(f)$ denotes its total degree, while $\mathsf{deg}_{x_i}(f)$ denotes its degree w.r.t. x_i. Let $f(\underline{x}) = f(x_1, .., x_n) \in \mathbb{R}[x_1, .., x_n]$ with $\mathsf{deg}_{x_k} f = d_k$, $k = 1, .., n$. If not specified, we denote $d = d(f) = \max\{d_1, .., d_n\}$.

We are interested in isolating the real roots of a system of polynomials $f_1(\underline{x}), .., f_s(\underline{x}) \in \mathbb{Z}[x_1, .., x_n]$, in a box $I_0 = [u_1, v_1] \times \cdots \times [u_n, v_n] \subset \mathbb{R}^n$, $u_k, v_k \in \mathbb{Q}$. We denote by $\mathcal{Z}_{\mathbb{K}^n}(f) = \{p \in \mathbb{K}^n; f(p) = 0\}$ the solution set in $\mathbb{K}^n$ of the equation $f(x) = 0$, where $\mathbb{K}$ is $\mathbb{R}$ or $\mathbb{C}$.

In what follows $\mathcal{O}_B$, resp. $\mathcal{O}$, means bit, resp. arithmetic, complexity and the $\widetilde{\mathcal{O}}_B$, resp. $\widetilde{\mathcal{O}}$, notation means that we are ignoring logarithmic factors. For $a \in \mathbb{Q}$, $\mathcal{L}(a) \geq 1$ is the maximum bit size of the numerator and the denominator. For a polynomial $f \in \mathbb{Z}[x_1, .., x_n]$, we denote by $\mathcal{L}(f)$ the maximum of the bitsize of its coefficients (including a bit for the sign). In the following, we will consider classes of polynomials such that $\log(d(f)) = \mathcal{O}(\mathcal{L}(f))$.

Also, to simplify the notation we introduce multi-indices, for the variable vector $\underline{x} = (x_1, .., x_n)$, $\underline{x}^{\underline{i}} := x_1^{i_1} \cdots x_n^{i_n}$, the sum $\sum_{\underline{i}=\underline{0}}^{\underline{d}} := \sum_{i_1=0}^{d_1} \cdots \sum_{i_n=0}^{d_n}$, and $\binom{\underline{d}}{\underline{i}} := \binom{d_1}{i_1} \cdots \binom{d_n}{i_n}$. The tensor Bernstein basis polynomials of multidegree degree $\underline{d}$ of a box I are denoted $B(\underline{x}; \underline{i}, \underline{d}; I) := B_{d_1}^{i_1}(x_1; u_1, u_1) \cdots B_{d_n}^{i_n}(x_n; u_n, u_n)$ where $I = [\underline{u}, \underline{v}] := [u_1, v_1] \times \cdots \times [u_n, v_n]$.

1.2 The general scheme

In this section, we describe the family of algorithms that we consider. The main ingredients are

- a suitable representation of the equations in a given (usually rectangular) domain, for instance a representation in the Bernstein basis or in the monomial basis;
- an algorithm to split the representation into smaller sub-domains;
- a reduction procedure to shrink the domain.

Different choices for each of these ingredients lead to algorithms with different practical behaviors. The general process is summarized in Alg. 1.

The instance of this general scheme that we obtain generalizes the continued fraction method for univariate polynomials; the realization of the main steps (b-e) can be summarized as follows:

b) Perform a precondition process and compute a lower bound on the roots of the system, in order to reduce the domain.

c) Apply interval analysis or sign inspection to identify if some f_i has constant sign in the domain, i.e. if the domain contains no roots.

d) Apply Miranda test to identify if the domain contains a single root. In this case output $(I, f_1, .., f_s)$.

e) If both tests fail, split the representation at $(1, .., 1)$ and continue.

Algorithm 1.1: Subdivision scheme

```
Input: A set of equations $f_1, f_2, .., f_s \in \mathbb{Z}[\underline{x}]$
       represented over a domain $I$.
Output: A list of disjoint domains, each containing one
        and only one real root of $f_1 = \cdots = f_s = 0$.
Initialize a stack $Q$ and add $(I, f_1, .., f_s)$ on top of it;
While $Q$ is not empty do
    a) Pop a system $(I, f_1, .., f_s)$ and:
    b) Perform a precondition process and/or a reduction
       process to refine the domain.
    c) Apply an exclusion test to identify if the domain
       contains no roots.
    d) Apply an inclusion test to identify if the domain
       contains a single root. In this case output
       $(I, f_1, .., f_s)$.
    e) If both tests fail split the representation into a
       number of sub-domains and push them to $Q$.
```

In the following sections, we are going to describe more precisely the specific steps and analyze their complexity. In Sec. 2, we describe the representation of domains via homographies and the connection with the Bernstein basis representation. Subdivision, based on shifts of univariate polynomials, reduction and preconditionning are analyzed in Sec. 3. Exclusion and inclusion tests as well as a generalization of Vincent's theorem to multivariate polynomials, are presented in Sec. 4. In Sec. 5, we recall the main properties of Continued Fraction expansion of real numbers and use them to analyze the complexity of a subdivision algorithm following this generic scheme. We conclude with examples produced by our C++ implementation in Sect. 6.

2. REPRESENTATION: HOMOGRAPHIES

A widely used representation of a polynomial f in a box is the tensor-Bernstein representation. De Casteljau's algorithm provides an efficient way to split this representation to smaller domains. A disadvantage is that converting integer polynomials to Bernstein form results in rational Bernstein coefficients. We follow an alternative approach that does not require basis conversion since it applies to monomial basis: We introduce a tensor-monomial representation, i.e. a representation in the monomial basis over $\mathbb{P}^1 \times \cdots \times \mathbb{P}^1$ and provide an algorithm to subdivide this representation analogously to the Bernstein case.

In a tensor-monomial representation a polynomial is represented as a tensor (higher dimensional matrix) of coefficients in the natural monomial basis, that is,

$$f(\underline{x}) = \sum_{i_1, .., i_n}^{d_1, .., d_n} c_{i_1 .. i_n} \underline{x}^{(i_1, .., i_n)} = \sum_{\underline{i}=\underline{0}}^{\underline{d}} c_{\underline{i}} \underline{x}^{\underline{i}},$$

for every equation f of the system. Splitting this representation is done using homographies. The main operation in this computation is the Taylor shift.

Definition 2.1. *A homography (or Mobius transformation) is a bijective projective transformation* $\mathcal{H} = (\mathcal{H}_1, .., \mathcal{H}_n)$, *defined over* $\mathbb{P}^1 \times \cdots \times \mathbb{P}^1$ *as*

$$x_k \mapsto \mathcal{H}_k(x_k) = \frac{\alpha_k x_k + \beta_k}{\gamma_k x_k + \delta_k}$$

with $\alpha_k, \beta_k, \gamma_k, \delta_k \in \mathbb{Z}$, $\gamma_k \delta_k \neq 0$, $k = 1, .., n$.

Using simple calculations, we can see that the inverse

$$\mathcal{H}_k^{-1}(x_k) = \frac{\delta_k x_k - \beta_k}{\gamma_k x_k - \alpha_k}$$

is also a homography. Also, notice that if $\det \mathcal{H} > 0$ then, taking proper limits when needed, we can write

$$\mathbb{R}_+ \mapsto \mathcal{H}_k(\mathbb{R}_+) = \left[\frac{\beta_k}{\delta_k}, \frac{\alpha_k}{\gamma_k}\right] \tag{1}$$

hence $H(f) : \mathbb{R}_+^n \to \mathbb{R}$,

$$H(f) := \prod_{k=1}^{n} (\gamma_k x_k + \delta_k)^{d_k} \cdot (f \circ \mathcal{H})(x)$$

is a polynomial defined over $\mathbb{R}_+^n$ that corresponds to the (possibly unbounded) box

$$I_H = \mathcal{H}(\mathbb{R}_+^n) = \left[\frac{\beta_1}{\delta_1}, \frac{\alpha_1}{\gamma_1}\right] \times \cdots \times \left[\frac{\beta_n}{\delta_n}, \frac{\alpha_n}{\gamma_n}\right], \tag{2}$$

of the initial system, in the sense that the zeros of the initial system in I_H are in one-to-one correspondence with the positive zeros of $H(f)$.

We focus on the computation of $H(f)$. We use the base homographies $T_k^c(f) = f|_{x_k = x_k + c}$ (translation by c) or simply $T_k(f)$ if $c = 1$, $C_k^c(f) = f|_{x_k = c x_k}$ (contraction by c) and $R_k(f) = x_k^{d_k} f|_{x_k = 1/x_k}$ (reciprocal polynomial). These notations are naturally extended to variable vectors; for instance $T^{\underline{c}} = (T_1^{c_1}, .., T_n^{c_n})$, $c = (c_1, .., c_n) \in \mathbb{Z}^n$. Complexity results for these computations appear in the following sections. We can see that they suffice to compute any homography:

Lemma 2.2. *The group of homographies* H *with coefficients in* $\mathbb{Z}$ *is generated by* R_k, C_k^c, T_k^c, $k = 1, .., n$, $c \in \mathbb{Z}$.

Proof. It can be verified that $H_k(f)$ is computed as

$$H_k(f) = C_k^{\gamma_k} R_k C_k^{\delta_k} T_k R_k C_k^{\beta_k/\delta_k - \alpha_k/\gamma_k} T_k^{\beta_k/\delta_k}(f)$$

where the product denotes composition. We abbreviate $C_k^{1/c} = R_k C_k^c R_k$ and $T_k^{1/c} = C_k^u T_k^1 R_k T_k^c R_k$. □

Representation via homography is not far from Bernstein representation:

Lemma 2.3. *Let* $f = \sum_{\underline{i}=0}^{\underline{d}} b_i \, B_i^n(\underline{x}, I_H)$ *the Bernstein expansion of* f *in the box* I_H *yielded by a homography* H. *If*

$$H(f) = C^\gamma R C^\delta T^1 R C^{\beta/\delta - \alpha/\gamma} T^{\beta/\delta}(f) = \sum_{\underline{i}=0}^{\underline{d}} c_i \underline{x}^{\underline{i}}$$

then $c_i = \binom{\underline{d}}{\underline{i}} \gamma^{\underline{i}} \delta^{\underline{d} - \underline{i}} b_i$.

Proof. Let $[u_k, v_k] = \left[\frac{\alpha_k}{\gamma_k}, \frac{\beta_k}{\delta_k}\right]$. For a tensor-Bernstein polynomial $\binom{\underline{d}}{\underline{i}} \frac{1}{(v-u)^d} (x-u)^i (v-x)^{d-i}$ we compute

$$C^\gamma R C^\delta T^1 R C^{v-u} T^u (\binom{\underline{d}}{\underline{i}} \frac{1}{(v-u)^d} (x-u)^i (v-x)^{d-i})$$

$$= C^\gamma R C^\delta R T^1 R C^{v-u} (\binom{\underline{d}}{\underline{i}} \frac{1}{(v-u)^d} x^i (v-u-x)^{d-i})$$

$$= C^\gamma R C^\delta R T^1 (\binom{\underline{d}}{\underline{i}} (x-1)^{d-i})$$

$$= C^\gamma R C^\delta (\binom{\underline{d}}{\underline{i}} x^i) = \binom{\underline{d}}{\underline{i}} \gamma^i \delta^{d-i} x^i$$

as needed. □

Corollary 2.4. *The Bernstein expansion of* f *in* I_H *is*

$$\sum_{i=0}^{d} \frac{c_i}{\binom{d}{i} \gamma^i \delta^{d-i}} B_i^d(\underline{x}; I_H).$$

That is, the coefficients of $H(f)$ *coincide with the Bernstein coefficients up to contraction and binomial factors.*

Thus tensor-Bernstein coefficients and tensor-monomial coefficients in a sub-domain of $\mathbb{R}_+^n$ differ only by multiplication by positive constant. In particular they are of the same sign. Hence this corollary allows us to take advantage of sign properties (eg. the variation diminishing property) of the Bernstein basis without computing it.

The resulting representation of the system consists of the transformed polynomials $H(f_1), .., H(f_n)$, represented as tensors of coefficients as well as $4n$ integers, $\alpha_k, \beta_k, \gamma_k, \delta_k$ for $k = 1, .., n$ from which we can recover the endpoints of the domain, using (2).

3. SUBDIVISION AND REDUCTION

3.1 The subdivision step

We describe the subdivision step using the homography representation. This is done at a point $\underline{u} = (u_1, .., u_n) \in \mathbb{Z}_{\geq 0}^n$. It consists in computing up to 2^n new sub-domains (depending on the number of nonzero u_k's), each one having $\underline{u}$ as a vertex.

Given $H(f_1), .., H(f_s)$ that represent the initial system at some domain, we consider the partition of $\mathbb{R}_+^n$ defined by the hyperplanes $x_k = u_k$, $k = 1, .., n$. These intersect at $\underline{u}$ hence we call this *partition at* $\underline{u}$. Subdividing at $\underline{u}$ is equivalent to subdividing the initial domain into boxes that share the common vertex $\mathcal{H}(\underline{u})$ and have faces either parallel or perpendicular to those of the initial domain.

We need to compute a homography representation for every domain in this partition. The computation is done coordinate wise; observe that for any domain in this partition we have, for all k, either $x_k \in [0, u_k]$ or $x_k \in [u_k, \infty]$. It suffices to apply a transformation that takes these domains to $\mathbb{R}_+$. In the former case, we apply $T_k^1 R_k C_k^{u_k}$ to the current polynomials and in the latter case we shift them by u_k, i.e. we apply $T_k^{u_k}$. The integers $\alpha_k, \beta_k, \gamma_k, \delta_k$ that keep track of the current domain can be easily updated to correspond to the new subdomain.

We can make this process explicit in general dimension: every computed subdomain corresponds to a binary number of length n, where the $k-$th bit is 1 if $T_k^1 R_k C^{u_k}$ is applied or 0 if $T_k^{u_k}$ is applied.

In our continued fraction algorithm the subdivision is performed at $\underline{u} = \underline{1}$.

Illustration. Let us illustrate this process in dimension two. The system f_1, f_2 is defined over $\mathbb{R}^2_{>0}$. We subdivide this domain into $[0,1]^2$, $[0,1] \times \mathbb{R}_{>1}$, $\mathbb{R}_{>1} \times [0,1]$ and $\mathbb{R}_{>1} \times \mathbb{R}_{>1}$. Equivalently, we compute four new pairs of polynomials, as illustrated in Fig. 1 (we abbreviate $S_k = T_k^1 R_k$).

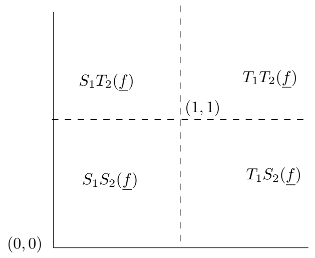

Figure 1: Subdividing the domain of $\underline{f}$.

Complexity of subdivision step. The transformation of a polynomial into two sub-domains, i.e. splitting w.r.t. one direction, consists of performing d^{n-1} univariate shifts, one for every coefficient $\in \mathbb{Z}[x_k]$ of $f \in \mathbb{Z}[x_k][x_1, .., \widehat{x_k}, .., x_n]$.

If the subdivision is performed in every direction, each transformation consists of d^{n-1} univariate shifts for every variable, i.e. nd^{n-1} shifts. There are 2^n sub-domains to compute, hence a total of $n^2 2^n d^{n-1}$ shifts have to be performed in a single subdivision step. We must also take into account that every time a univariate shift is performed, the coefficient bitsize increases.

The operations

$$T_k(f) = f|_{x_k = x_k+1} \text{ and } T_k R_k(f) = (x_k+1)^{d_k} f|_{x_k = \frac{1}{x_k+1}}$$

are essentially of the same complexity, except that the second requires one to exchange the coefficient of $c_{i_1,..,i_k,..,i_n}$ with $c_{i_1,..,d_k - i_k,..,i_n}$ before translation, i.e. an additional $\mathcal{O}(d^n)$ cost. Hence we only need to consider the case of shifts for the complexity.

The continued fraction algorithm subdivides a domain using unit shifts and inversion. Successive operations of this kind increase the bitsize equivalently to a big shift by the sum of these units. Thus it suffices to consider the general computation of $f(\underline{x} + \underline{u})$ to estimate the complexity of the subdivision step.

Lemma 3.1 (Shift complexity). *The computation of $f(\underline{x} + \underline{u})$ with $\mathcal{L}(f) = \tau$ and $\mathcal{L}(u_k) \le \sigma$, $k = 1, .., n$ can be performed in $\widetilde{\mathcal{O}}_B(n^2 d^n \tau + d^{n+1} n^3 \sigma)$.*

Proof. We use known facts for the computation of $T_k^{u_k}(f)$ for univariate polynomials. If $\deg_k f = d_k$ and f is univariate, this operation is performed in $\widetilde{\mathcal{O}}_B(d_k^2 \sigma + d_k \tau)$; the resulting coefficients are of bitsize $\tau + d_k \sigma$ [20]. Hence $f(x_1, .., x_k + u_k, .., x_n)$ is computed in $\widetilde{\mathcal{O}}_B(d^{n-1}(d_k^2 \sigma + d_k \tau))$.

Suppose we have computed $f(x_1 + u_1, x_{k-1} + u_{k-1}, x_k, .., x_n)$ for some k. The coefficients are of bitsize $\tau + \sum_{i=1}^{k-1} \sigma_i$. The computation of shift w.r.t. $k-$th variable $f(x_1 + u_1, .., x_k + u_k, x_{k+1}, .., x_n)$ results in a polynomial of bitsize $\tau + \sum_{i=1}^{k} \sigma_i$ and consists of $d^{n-1} \widetilde{\mathcal{O}}_B(d^2 \sum_{i=1}^{k} \sigma_i + d\tau))$ operations. That is, we perform d^{n-1} univariate polynomial shifts, one for every coefficient of f in $\mathbb{Z}[x_k][x_1, .., \widehat{x_k}, .., x_n]$.

This gives a total cost for computing $f(\underline{x} + \underline{u})$ of

$$d^{n-1} \sum_{k=1}^{n} \left(d^2 \sum_{i=1}^{k} \sigma_i + d\tau \right) = nd^n \tau + d^{n+1} \sum_{k=1}^{n} (n+1-k) \sigma_k.$$

The latter sum implies that it is faster to apply the shifts with increasing order, starting with the smallest number u_k. Since $\sigma_k = \mathcal{O}(\sigma)$ for all k, and we must shift a system of $\mathcal{O}(n)$ polynomials we obtain the stated result. $\square$

Let us present an alternative way to compute a sub-domain using contraction, preferable when the bitsize of $\underline{u}$ is big. The idea behind this is the fact that T_k^c and $T_k^1 C^c$ compute the same sub-domain, in two different ways.

Lemma 3.2. *If $f = \sum_{\underline{i}=\underline{0}}^{\underline{d}} c_{\underline{i}} \underline{x}^{\underline{i}}$, $\mathcal{L}(f) = \tau$, then the coefficients of $C^u(f)$, $\mathcal{L}(u_k) \le \sigma$, $k = 1, .., n$, can be computed in $\widetilde{\mathcal{O}}_B(d^n \tau + nd^{n+1} \sigma)$.*

Proof. The operation, i.e. computing the new coefficients $c_{\underline{i}} \underline{u}^{\underline{i}}$ can be done with $\widetilde{\mathcal{O}}(d^n)$ multiplications: Since $\underline{u}^{(i_1,..,i_k,..,i_n)} = u_k \underline{u}^{(i_1,..,i_k - 1,..,i_n)}$, if these powers are computed successively then every coefficient is computed using two multiplications. Moreover, it suffices to keep in memory the n powers $u^{(i_1,..,i_{k-1}, i_k - 1, i_{k+1},..,i_n)}$, $k = 1, .., n$ in order to compute any $\underline{u}^{\underline{i}} c_{\underline{i}}$. Geometrically this can be understood as a stencil of n points that sweeps the coefficient tensor and updates every element using one neighbor at each time. The bitsize of the multiplied numbers is $\mathcal{O}(\tau + d\sigma)$ hence the result follows. $\square$

Now if we consider a contraction followed by a shift by 1 w.r.t. x_k for $\mathcal{O}(n)$ polynomials we obtain $\widetilde{\mathcal{O}}_B(n^2 d^n \tau + n^3 d^{n+1} + nd^{n+1} \sigma)$ operations for the computation of the domain. The disadvantage is that the resulting coefficients are of bitsize $\mathcal{O}(\tau + d\sigma)$ instead of $\mathcal{O}(\tau + n\sigma)$ with the use of shifts. Also note that this operation would compute a expansion of the real root which differs from continued fraction expansion.

3.2 Reduction: Bounds on the range of f

In this section we define univariate polynomials whose graph in $\mathbb{R}^{n+1}$ bounds the graph of f. For every direction k, we provide two polynomials bounding the values of f in $\mathbb{R}^n$ from below and above respectively.

Define

$$m_k(f; x_k) = \sum_{i_k=0}^{d_k} \min_{i_1,..,\widehat{i_k},..,i_n} c_{i_1..i_n} x_k^{i_k} \tag{3}$$

$$M_k(f; x_k) = \sum_{i_k=0}^{d_k} \max_{i_1,..,\widehat{i_k},..,i_n} c_{i_1..i_n} x_k^{i_k} \tag{4}$$

Lemma 3.3. *For any $x \in \mathbb{R}^n_+$, $n > 1$ and any $k = 1, .., n$, we have*

$$m_k(f; x_k) \le \frac{f(x)}{\prod_{s \ne k} \sum_{i_s=0}^{d_s} x_s^{i_s}} \le M_k(f; x_k). \tag{5}$$

PROOF. For $x \in \mathbb{R}_+^n$, we can directly write

$$f(x) \le \left(\sum_{i_k=0}^{d_k} \max_{i_1,..,\widehat{i_k},..,i_n} c_{i_1..i_n} x_k^{i_k} \right) \prod_{s \neq k} \sum_{i_s=0}^{d_s} x_s^{i_s}$$

The product of power sums is greater than 1; divide both sides by it. Analogously for $M_k(f; x_k)$. □

COROLLARY 3.4. *Given $k \in \{1,..,n\}$, if $u \in \mathbb{R}_+^n$ with $u_k \in]0, \mu_k]$, where*

$$\mu_k = \begin{cases} \text{min. pos. root of } M_k(f, x_k) & \text{if } M_k(f; 0) < 0 \\ \text{min. pos. root of } m_k(f, x_k) & \text{if } m_k(f; 0) > 0 \\ 0 & \text{otherwise} \end{cases},$$

then $f(u) \neq 0$. Consequently, all positive roots of f lie in $\mathbb{R}_{>\mu_1} \times \cdots \times \mathbb{R}_{>\mu_n}$. Also, for $u \in \mathbb{R}_+^n$ with $u_k \in [\mathcal{M}_k, \infty]$,

$$\mathcal{M}_k = \begin{cases} \text{max. pos. root of } M_k(f, x_k) & \text{if } M_k(f; \infty) < 0 \\ \text{max. pos. root of } m_k(f, x_k) & \text{if } m_k(f; \infty) > 0 \\ \infty & \text{otherwise} \end{cases},$$

it is $f(u) \neq 0$, i.e. all pos. roots are in $\mathbb{R}_{<\mathcal{M}_1} \times \cdots \times \mathbb{R}_{<\mathcal{M}_n}$.

Combining both bounds we deduce that $[\mu_1, \mathcal{M}_1] \times \cdots \times [\mu_n, \mathcal{M}_n]$ is a bounding box for $f^{-1}(\{0\}) \cap \mathbb{R}_+^n$.

PROOF. The denominator in (5) is always positive in $\mathbb{R}_+^n$. Let $\underline{u} \in \mathbb{R}^n$ with $u_k \in [0, \mu_k]$. If $M_k(f, 0) < 0$ then also $M_k(f, u) < 0$ and it follows $f(\underline{u}) < 0$. Similarly $m_k(f, 0) > 0 \Rightarrow m_k(f, u) > 0 \Rightarrow f(\underline{u}) < 0$. The same arguments hold for $[\mathcal{M}_k, \infty]$, $M_k(f; \infty) = R(M_k(f; x_k))(0)$, $m_k(f; \infty) = R(m_k(f; x_k))(0)$, and $R(f)$, since lower bounds on the zeros of $R(f)$ yield upper bounds on the zeros of f. □

Thus lower and upper bounds on the $k-$th coordinates of the roots of $(f_1, .., f_s)$ are given by

$$\max_{i=1,..,s} \{\mu_k(f_i)\} \quad \text{and} \quad \min_{i=1,..,s} \{\mathcal{M}_k(f_i)\} \tag{6}$$

respectively, i.e. the intersection of these bounding boxes.

We would like to remain in the ring of integers all along the process, thus integer lower or upper bounds will be used.These can be the floor or ceil of the above roots of univariate polynomials, or even known bounds for them, e.g. Cauchy's bound.

If the minimum and maximum are taken with the ordering of coefficients defined as $c_i \prec c_j \iff c_i \binom{d}{j} \gamma^j \delta^{d-j} < c_j \binom{d}{i} \gamma^i \delta^{d-i}$ then different $m_k(f, x_k), M_k(f, x_k)$ polynomials are obtained. By Cor. 2.4 their control polygon is the lower and upper hull respectively of the projections of the tensor-Bernstein coefficients to the $k-$th direction and are known to converge quadratically to simple roots when preconditioning (described in the following paragraph) is utilized [12, Cor. 5.3].

Complexity analysis. The analysis of the subdivision step in Sect. 3.2 applies as well to the reduction step, since reducing the domain means computing a new subdomain and ignoring the remaining part.

If a lower bound $\underline{l}$ is known, with $\mathcal{L}(l_k) = \widetilde{\mathcal{O}}(\sigma)$, then the reduction step is performed in $\widetilde{\mathcal{O}}_B(n^2 d^n \tau + d^{n+1} n^3 \sigma)$. This is an instance of Lem. 3.1.

The projections of Lem. 3.3 are computed using $\mathcal{O}(d^n)$ comparisons. The computation of $\underline{l}$ costs $\widetilde{\mathcal{O}}_B(d^3\tau)$ in average, for solving these projections using univariate CF algorithm. Another option would to compute well known lower bounds on their roots, for instance Cauchy's bound in $\mathcal{O}(d)$.

Illustration. Consider a bi-quadratic $f_0 \in \mathbb{R}[x, y]$, namely, $\deg_{x_1} f_0 = \deg_{x_2} f_0 = 2$ with coefficients c_{ij}. Suppose that $f_0 = H(f)$ for $I_0 = I_H$. We compute

$$m_1(f, x_1) = \sum_{i=0}^{2} \min_{j=0,..2} c_{ij}\, x^i \quad \text{and} \quad M_1(x) = \sum_{i=0}^{2} \max_{j=0,..2} c_{ij}\, x^i.$$

thus $m(x) \le \frac{f(x_1,x_2)}{1+x_2+x_2^2} \le M(x)$. Fig. 2 shows how these univariate quadratics bound the graph of f in I_0.

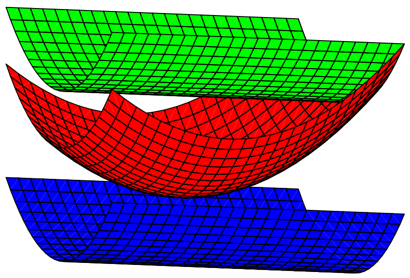

Figure 2: The enveloping polynomials $M_1(x)$, $m_1(x)$ in domain I_0 for a bi-quadratic polynomial $f(x, y)$.

3.3 Preconditioning

To improve the reduction step, we use preconditioning. The aim of a preconditioner is to tune the system so that it can be tackled more efficiently; in our case we aim at improving the bounds of Cor. 3.4.

A preconditioning matrix P is an invertible $s \times s$ matrix that transforms a system $(f_1, .., f_s)^t$ into the equivalent one $P \cdot (f_1, .., f_s)^t$. This transformation does not alter the roots of the system, since the computed equations generate the same ideal. The bounds obtained on the resulting system can be used directly to reduce the domain of the equations before preconditioning. Preconditioning can be performed to a subset of the equations.

Since we use a reduction process using Cor. 3.4 we want to have among our equations n of them whose zero locus $f^{-1}(\{0\})$ is orthogonal to the $k-$th direction, for all k.

Assuming a square system, we precondition $H(f_1), .., H(f_n)$ to obtain a locally orthogonal to the axis system; an ideal preconditioner would be the Jacobian of the system evaluated at a common root; instead, we evaluate $J_{H(f)}$ at the image of the center $\underline{u}$ of the initial domain I_H, $u_k = \frac{\alpha_k \delta_k + \beta_k \gamma_k}{2\gamma_k \delta_k}$. Thus we must compute the inverse of the Jacobian matrix $J_{H(\underline{f})}(x) = [\partial_{x_i} H(f_j)(x)]_{1 \le i,j, \le n}$ evaluated at $\underline{u}' := \mathcal{H}(\underline{u}) = (\delta_1/\gamma_1, .., \delta_n/\gamma_n)$.

Precondition step complexity. Computing $J_{H(f)}(\underline{u}) \cdot (H(f_1), .., H(f_n))^t$ is done with cost $\widetilde{\mathcal{O}}_B(n^2 d^n)$ and evaluating at u' has cost $\widetilde{\mathcal{O}}_B(n^2 d^{n-1})$. We also need $\widetilde{\mathcal{O}}_B(n^2)$ for inversion and $\mathcal{O}(n^2 d^n)$ for multiplying polynomials times scalar as well as summing polynomials. This gives a precondition cost of order $\mathcal{O}(n^2 d^n)$.

4. EXCLUSION – INCLUSION CRITERIA

A subdivision scheme is able to work when two tests are available: one that identifies empty domains (exclusion test) and one that identifies domains with exactly one zero (inclusion test). If these two tests are negative, a domain cannot be neither included nor excluded so we need to apply further reduction/subdivision steps to it. The certification is the following: if the result of the test is affirmative, then this is undoubtedly true.

Exclusion test. The bounding functions defined in the previous section provide a fast filter to exclude empty domains. Define $\min\{\} = \infty$ and $\max\{\} = 0$.

COROLLARY 4.1. *If for some $k \in \{1, .., n\}$ and for some $i \in \{1, .., s\}$ it is $\mu_k(f_i) = \infty$ or $\mathcal{M}_k(f_i) = 0$ then the system has no solutions. Also, if $\max_{i=1,..,s}\{\mu_k(f_i)\} > \min_{i=1,..,s} \{\mathcal{M}_k(f_i)\}$ then there can be no solution to the system.*

PROOF. For the former statement observe that f_i has no real positive roots, thus the system has no roots. The latter statement means that the reduced domains of each f_i, $i = 1, .., s$ do not intersect, thus there are no solutions. $\Box$

We can use interval arithmetic to identify additional empty domains; if the sign of some initial f_i is constant in $I_H = \mathcal{H}(\mathbb{R}^n_{>0})$ then this domain is discarded. We can also simply inspect the coefficients of each $H(f_i)$; if there are no sign changes then there corresponding box contains no solution.

The accuracy of these criteria greatly affects the performance of the algorithm. In particular, the sooner an empty domain is rejected the less subdivisions will take place and the process will terminate faster. We justify that the exclusion criteria will eventually succeed on an empty domain by proving a generalization of Vincent's theorem to the tensor multivariate case.

THEOREM 4.2. *Let $f(\underline{x}) = \sum_{\underline{i}=\underline{0}}^{\underline{d}} c_{\underline{i}}\, \underline{x}^{\underline{i}}$ be a polynomial with real coefficients, such that it has no (complex) solution with $\Re(z_k) \geq 0$ for $k = 1, .., n$. Then all its coefficients $c_{i_1,..,i_n}$ are of the same sign.*

PROOF. We prove the result by induction on n, the number of variables. For $n = 1$, this is the classical Vincent's theorem [1].

Consider now a polynomial

$$f(x_1, x_2) = \sum_{0 \leq i_1 \leq d_1, 0 \leq i_2 \leq d_2} c_{i_1,i_2}\, x_1^{i_1}\, x_2^{i_2}$$

in two variables with no (complex) solution such that $\Re(x_i) \geq 0$ for $i = 1, 2$. We prove the result for $n = 2$, by induction on the degree $d = d_1 + d_2$. The property is obvious for polynomials of degree $d = 0$. Let us assume it for polynomials of degree less than d.

By hypothesis, for any $z_1 \in \mathbb{C}$ with $\Re(z_1) \geq 0$, the univariate polynomial $f(z_1, x_2)$ has no root with $\Re(x_2) \geq 0$. According to Lucas theorem [11], the complex roots of $\partial_{x_2} f(z_1, x_2)$ are in the convex hull of the complex roots of $f(z_1, x_2)$. Thus, there is no root of $\partial_{x_2} f(x_1, x_2)$ with $\Re(x_1) \geq 0$ and $\Re(x_2) \geq 0$. By induction hypothesis, the coefficients of $\partial_{x_2} f(x_1, x_2)$ are of the same sign. We decompose P as

$$f(x_1, x_2) = f(x_1, 0) + f_1(x_1, x_2)$$

where $f_1(x_1, x_2) = \sum_{0 \leq i_1 \leq d_1, 1 \leq i_2 \leq d_2} c_{i_1,i_2}\, x_1^{i_1}\, x_2^{i_2}$ with c_{i_1,i_2} of the same sign, say positive. By Vincent theorem in one variable, as $f(x_1, 0)$ has no root with $\Re(x_1) \geq 0$, the coefficients $c_{i_1,0}$ of $f(x_1, 0)$ are also of the same sign. If this sign is different from the sign of c_{i_1,i_2} for $i_2 \geqslant 1$ (ie. negative here), then $f(0, x_2)$ has one sign variation in its coefficients list. By Descartes rule, it has one real positive root, which contradicts the hypothesis on f. Thus, all the coefficients have the same sign.

Assume that the property has been proved for polynomials in $n - 1$ variables and let us consider a polynomial $f(\underline{x}) = \sum_{i=0}^{d} c_i\, \underline{x}^i$ in n variables with no (complex) solution such that $\Re(x_k) \geq 0$ for $k = 1, .., n$. For any $z_1, .., z_{n-1} \in \mathbb{C}$ with $\Re(z_k) \geq 0$, for $k = 1, .., n - 1$, the polynomial $f(z_1, .., z_{n-1}, x_n)$ and $\partial_{x_n} f(z_1, .., z_{n-1}, x_n)$ has no root with $\Re(x_n) \geq 0$. By Lucas theorem and induction hypothesis on the degree, $\partial_{x_n} f(\underline{x})$ has coefficients of the same sign. We also have $f(x_1, .., x_{n-1}, 0)$ with coefficients of the same sign, by induction hypothesis on the number of variables. If the two signs are different, then $f(0, .., 0, x_n)$ has one sign variation in its coefficients and thus one real positive root, say ζ_n, which cannot be the case, since $(0, .., 0, \zeta_n)$ would yield a real root of f. We deduce that all the coefficients of f are of the same sign.

This completes the induction proof of the theorem. $\Box$

This implies that empty regions will be eventually excluded by sign inspection.

COROLLARY 4.3. *Let $H(f) = \sum_{\underline{i}=0}^{\underline{d}} c_{\underline{i}}\, \underline{x}^{\underline{i}}$ be the representation of f through $\mathcal{H}$ in a box $I_H = [\underline{u}, \underline{v}]$. If there is no toot $\underline{z} \in \mathbb{C}^n$ of f such that*

$$\left| z_k - \frac{u_k + v_k}{2} \right| \leq \frac{v_k - u_k}{2}, \text{ for } k = 1, .., n,$$

then all the coefficients $c_{i_1,..,i_n}$ are of the same sign.

That is, if $dist_\infty(\mathcal{Z}_{\mathbb{C}^n}(f), m) > \varepsilon$, where m is the center of I_H, then I_H is excluded by sign conditions.

PROOF. The interval $[u_k, v_k]$ is transformed by $\mathcal{H}^{-1}$ into $[0, +\infty]$ and the disk $\left|z_k - \frac{u_k+v_k}{2}\right| \leq \frac{v_k-u_k}{2}$ is transformed into the half complex plane $\Re(z_k) \geq 0$. We deduce that $H(f)$ has no root with $\Re(z_k) \geq 0$, $k = 1, .., n$. By Thm. 4.2, the coefficients of $H(f)$ are of the same sign. $\Box$

We deduce that if a domain is far enough from the zero locus of some f_i then it will be excluded, hence redundant empty domains concentrate only in a neighborhood of $\underline{f} = \underline{0}$.

DEFINITION 4.4. *The tubular neighborhood of size ε of f_i is the set*

$$\tau_\varepsilon(f_i) = \{x \in \mathbb{R}^n \;:\; \exists z \in \mathbb{C}^n,\ f_i(z) = 0,\ s.t.\ \|z - x\|_\infty < \varepsilon\}.$$

We bound the number of boxes that are not excluded at each level of the subdivision tree.

LEMMA 4.5. *Assume that for $\varepsilon_0 > 0$, $\cap_i \tau_{\varepsilon_0}(f_i) \cap I_0$ is bounded. Then the number of boxes of size $\varepsilon < \varepsilon_0$ kept by the algorithm is less than $(1 + \frac{\sqrt{n}}{2})^n\, c$, where $c > 0$ is such that $\forall \varepsilon$ st. $\varepsilon_0 > \varepsilon > 0$,*

$$V(f, \varepsilon) := \text{volume}\left(\cap_{i=1}^s \tau_\varepsilon(f_i) \cap I_0)\right) \leq c\, \epsilon^n.$$

PROOF. Consider a subdivision of a domain I_0 into boxes of size $\varepsilon < \varepsilon_0$. We will bound the number N of boxes in this subdivision that are not rejected by the algorithm. By Cor. 4.3 if a box is not rejected, then we have for all $i = 1, .., s$ $\text{dist}_\infty(\mathcal{Z}_{\mathbb{C}^n}(f_i), m) < \varepsilon$, where m is the center of the box. Thus all the points of this box are at distance $< \varepsilon(1 + \frac{\sqrt{n}}{2})$ to $\mathcal{Z}_{\mathbb{C}^n}(f_i)$ that is in $\cap_{i=1}^s \tau_{\varepsilon(1+\frac{\sqrt{n}}{2})}(f_i) \cap I_0$.

To bound N, it suffices to estimate the n-dimensional volume $V(f, \varepsilon)$, since we have:

$$N \varepsilon^n \leq \text{volume}\left(\cap_{i=1}^s \tau_{\varepsilon(1+\frac{\sqrt{n}}{2})}(f_i) \cap I_0\right) = V(f, \varepsilon\, (1 + \frac{\sqrt{n}}{2})).$$

When ε tends to 0, this volume becomes equivalent to a constant times ε^n. For a square system with single roots in I_0, it becomes equivalent to the sum for all real roots ζ in I_0 of the volumes of parallelotopes in n dimensions of height 2ε and unitary edges proportional to the gradients of the polynomials evaluated at the common root; It is thus bounded by $\varepsilon^n 2^n \sum_{\zeta \in I_0} \frac{|J_f(\zeta)|}{\prod_i ||\nabla f_i(\zeta)||}$. We deduce that there exists a constant $c \geq 2^n \sum_{\zeta \in I_0} \frac{|J_f(\zeta)|}{\prod_i ||\nabla f_i(\zeta)||}$ such that $V(f, \varepsilon) \leq c\,\varepsilon^n < \infty$. For overdetermined systems, the volume is bounded by a similar expression. Since $V(f, \varepsilon)\varepsilon^{-n}$ has a limit when ε tends to 0, we deduce the existence of the finite constant c and the bound of the lemma on the number of kept boxes of size ε. $\square$

Inclusion test. We present a test that discovers common solutions, in a box, or equivalently in $\mathbb{R}^n_+$, through homography. To simplify the statements we assume that the system is square, i.e. $s = n$.

Definition 4.6. *The* lower face *polynomial of f w.r.t. direction k is* $\text{low}(f, k) = f|_{x_k=0}$. *The* upper face *polynomial of f w.r.t. k is* $\text{upp}(f, k) = f|_{x_k=\infty} := R_k(f)|_{x_k=0}$.

Lemma 4.7 (Miranda Theorem [21]). *If for some permutation $\pi : \{1, .., n\} \rightarrow \{1, .., n\}$, $\text{sign}(\text{low}(H(f_k), \pi(k)))$ and $\text{sign}(\text{upp}(H(f_k), \pi(k)))$ are constant and opposite for all $k = 1, .., n$, then the equations $(f_1, .., f_n)$ have at least one root in I_H.*

The implementation of the Miranda test can be done efficiently if we compute a $0-1$ matrix with $(i, j)-$th entry 1 iff $\text{sign}(\text{low}(H(f_i), j))$ and $\text{sign}(\text{upp}(H(f_i), j))$ are opposite. Then, Miranda test is satisfied iff there is no zero row and no zero column. To see this observe that the matrix is the sum of a permutation matrix and a $0-1$ matrix iff this permutation satisfies Miranda's test.

Combined with the following simple fact, we have a test that identifies boxes with a single root.

Lemma 4.8. *If $\det J_f(x)$ has constant sign in a box I, then there is at most one root of $f = (f_1, .., f_n)$ in I.*

Proof. Suppose $u, v \in I$ are two distinct roots; by the mean value theorem there is a point w on the line segment $\overline{uv}$, and thus in I, s.t. $J_f(w) \cdot (u - v) = f(u) - f(v) = \mathbf{0}$ hence $\det J_f(w) = 0$. $\square$

Complexity of the inclusion criteria. Miranda test can be decided with $\mathcal{O}(n^2)$ evaluations on interval (cf. [9]) as well as one evaluation of $J_{\underline{f}}$, overall $\mathcal{O}(n^2 d^n)$ operations. The cost of the inclusion test is dominated by the cost of evaluating $\mathcal{O}(n)$ polynomials of size $\mathcal{O}(d^n)$ on an interval, i.e. $\mathcal{O}(nd^n)$ operations suffice.

Proposition 4.9. *If the real roots of the square system in the initial domain I_0 are simple, then Alg. 1 stops with boxes isolating the real roots in I_0.*

Proof. If the real roots of $f = (f_1, .., f_n)$ in I_0 are simple, in a small neighborhood of them the Jacobian of f has a constant sign. By the inclusion test, any box included in this neighborhood will be output if and only if it contains a single root and has no real roots of the jacobian. Otherwise, it will be further subdivided or rejected. Suppose that the subdivision algorithm does not terminate. Then the size of the boxes kept at each step tends to zero. By Cor. 4.3, these boxes are in the intersection of the tubular neighborhoods $(\cap_{i=1}^s \text{tub}_\varepsilon(f_i)) \cap \mathbb{R}^n$ for $\varepsilon > 0$ the maximal size of the kept boxes. If ε is small enough, these boxes are in a neighborhood of a root in which the Jacobian has a constant size, hence the inclusion test will succeed. By the exclusion criteria, a box domain is not subdivided indefinitely, but is eventually rejected when the coefficients become positive. Thus the algorithm either outputs isolating boxes that contains a real root of the system or rejects empty boxes. This shows, by contradiction, the termination of the subdivision algorithm. $\square$

5. THE COMPLEXITY OF MCF

In this section we compute a bound on the complexity of the algorithm that exploits the continued fraction expansion of the real roots of the system. Hereafter, we call this algorithm MCF (Multivariate Continued Fractions). Since the analysis of the reduction steps of Sec. 3 and the Exclusion-Inclusion test of Sec. 4 would require much more developments, we simplify the situation and analyze a variant of this algorithm. We assume that two oracles are available. One that computes, exactly, the partial quotients of the positive real roots of the system, and one that counts exactly the number of real roots of the system inside a hypercube in the open positive orthant, namely $\mathbb{R}^n_+$. In what follows, we will assume the cost of the first oracle is bounded by $\mathcal{C}_1$, while the cost of the second is bounded by $\mathcal{C}_2$, and we derive the total complexity of the algorithm with respect to these parameters. In any case the number of reduction or subdivision steps that we derive is a lower bound on the number of steps that every variant of the algorithm will perform. The next section presents some preliminaries on continued fractions, and then we detail the complexity analysis.

5.1 About continued fractions

Our presentation follows closely [7]. For additional details we refer the reader to, e.g., [22, 3, 19]. In general a *simple (regular) continued fraction* is a (possibly infinite) expression of the form

$$c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + ..}} = [c_0, c_1, c_2, ..],$$

where the numbers c_i are called *partial quotients*, $c_i \in \mathbb{Z}$ and $c_i \geq 1$ for $i > 0$. Notice that c_0 may have any sign, however, in our real root isolation algorithm $c_0 \geq 0$, without loss of generality. By considering the recurrent relations

$$\begin{array}{lll} P_{-1} = 1, & P_0 = c_0, & P_{n+1} = c_{n+1} P_n + P_{n-1}, \\ Q_{-1} = 0, & Q_0 = 1, & Q_{n+1} = c_{n+1} Q_n + Q_{n-1}, \end{array}$$

it can be shown by induction that $R_n = \frac{P_n}{Q_n} = [c_0, c_1, .., c_n]$, for $n = 0, 1, 2, ...$

If $\gamma = [c_0, c_1, ..]$ then $\gamma = c_0 + \frac{1}{Q_0 Q_1} - \frac{1}{Q_1 Q_2} + .. = c_0 + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{Q_{n-1} Q_n}$ and since this is a series of decreasing alternating terms it converges to some real number γ. A finite section $R_n = \frac{P_n}{Q_n} = [c_0, c_1, .., c_n]$ is called the $n-$th *convergent* (or *approximant*) of γ and the tails $\gamma_{n+1} = [c_{n+1}, c_{n+2}, ..]$ are known as its *complete quotients*. That is $\gamma = [c_0, c_1, .., c_n, \gamma_{n+1}]$ for $n = 0, 1, 2, ...$ There is an one to one correspondence between the real numbers and the continued fractions, where evidently the finite continued fractions correspond to rational numbers.

It is known that $Q_n \geq F_{n+1}$ and that $F_{n+1} < \phi^n < F_{n+2}$, where F_n is the $n-$th Fibonacci number and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Continued fractions are the best rational approximation(for a given denominator size). This is as follows:

$$\frac{1}{Q_n(Q_{n+1}+Q_n)} \leq \left|\gamma - \frac{P_n}{Q_n}\right| \leq \frac{1}{Q_n Q_{n+1}} < \phi^{-2n+1}. \quad (7)$$

Let $\gamma = [c_0, c_1, ..]$ be the continued fraction expansion of a real number. The Gauss-Kuzmin distribution [3] states that for almost all real numbers γ (meaning that the set of exceptions has Lebesgue measure zero) the probability for a positive integer δ to appear as an element c_i in the continued fraction expansion of γ is

$$Prob[c_i = \delta] \backsimeq \lg \frac{(\delta+1)^2}{\delta(\delta+2)}, \quad \text{for any fixed } i > 0. \quad (8)$$

The Gauss-Kuzmin law induces that we can not bound the mean value of the partial quotients or in other words that the expected value (arithmetic mean) of the partial quotients is diverging, i.e.

$$E[c_i] = \sum_{\delta=1}^{\infty} \delta \, Prob[c_i = \delta] = \infty, \text{ for } i > 0.$$

Surprisingly enough the geometric (and the harmonic) mean is not only asymptotically bounded, but is bounded by a constant, for almost all $\gamma \in \mathbb{R}$. For the geometric mean this is the famous Khintchine's constant [10], i.e.

$$\lim_{n\to\infty} \sqrt[n]{\prod_{i=1}^{n} c_i} = \mathcal{K} = 2.685452001...$$

It is not known if $\mathcal{K}$ is a transcendental number. The expected value of the bit size of the partial quotients is a constant for almost all real numbers, when $n \to \infty$ or n sufficiently big [10]. Notice that in (8), $i > 0$, thus $\gamma \in \mathbb{R}$ is uniformly distributed in $(0, 1)$. Let $\mathcal{L}(c_i) \triangleq b_i$, then

$$E[b_i] = \mathcal{O}(\lg \mathcal{K}) = \mathcal{O}(1). \quad (9)$$

5.2 Complexity results

Let σ be an upper bound on the bitsize of the partial quotient that appear during the execution of the algorithm.

Lemma 5.1. *The number of reduction and subdivision steps that the algorithm performs is* $\widetilde{\mathcal{O}}(n^2\tau d^{2n-1})$.

Proof. Let $\zeta = (\zeta_1, .., \zeta_n)$ be a real root of the system. It suffices to consider the number of steps needed to isolate the i coordinate of ζ.

Recall, that we assume, working in the positive orthant, we can compute exactly the next partial quotient in each coordinate; in other words a vector $l = (l_1, .., l_n)$, where each l_i, $1 \leq i \leq n$, is the partial quotient of a coordinate of a positive real[1] solution of the system.

Let $k_i(\zeta)$ be the number of steps needed to isolate the i coordinate of the real root ζ. The analysis is similar to the univariate case. The successive approximations of ζ_i by the lower bound l_i, yield the $k_i(\zeta)$-th approximant, $\frac{P_{k_i(\zeta)}}{Q_{k_i(\zeta)}}$ of ζ_i, which using (7) satisfies

$$\left|\frac{P_{k_i(\zeta)}}{Q_{k_i(\zeta)}} - \zeta_i\right| \leq \frac{1}{Q_{k_i(\zeta)}Q_{k_i(\zeta)+1}} < \phi^{-2k_i(\zeta)+1}.$$

In order to isolate ζ_i, it suffices to have

$$\left|\frac{P_{k_i(\zeta)}}{Q_{k_i(\zeta)}} - \zeta_i\right| \leq \Delta_i(\zeta),$$

where $\Delta_i(\zeta)$ is the local separation bound of ζ_i, that is the smallest distance between ζ_i and all the other i-coordinates of the positive real solutions of the system.

Combining the last two equations, we deduce that to achieve the desired approximation, we should have $\phi^{-2k_i(\zeta)+1} \leq \Delta_i(\zeta)$, or $k_i(\zeta) \geq \frac{1}{2} - \frac{1}{2}\lg \Delta_i(\zeta)$. That is to isolate the i coordinate it suffices to perform $\mathcal{O}(-\frac{1}{2}\lg \Delta_i(\zeta))$ steps. To compute the total number of steps, we need to sum over all positive real roots and multiply by n, which is the number of coordinates, that is

$$n \sum_{\zeta \in V} k_i(\zeta) \leq n\frac{1}{2}R - n\frac{1}{2}\sum_{\zeta\in V} \lg \Delta_i(\zeta) = n\frac{1}{2}R - n\frac{1}{2}\lg \prod_{\zeta \in V} \Delta_i(\zeta),$$

where $|V| = R$ is the number of positive real roots.

To bound the logarithm of the product, we use $\texttt{DMM}_n$ [8], i.e. aggregate separation bounds for multivariate, zero-dimensional polynomial systems. It holds

$$\begin{array}{lcl} \prod_{\zeta\in V} \Delta_i(\zeta) & \geq & 2^{-2n\tau d^{2n-1} - d^{2n}/2} \, (nd^n)^{-nd^{2n}} \\ -\log \prod_{\zeta\in V} \Delta_i(\zeta) & \leq & 2n\tau d^{2n-1} + 2nd^n \lg(nd^{2n}). \end{array}$$

Taking into account that $R \leq d^n$ we conclude that the number of steps is $\widetilde{\mathcal{O}}(n^2\tau d^{2n-1})$. □

Proposition 5.2. *The total complexity of the algorithm is* $\widetilde{\mathcal{O}}_B(2^n n^7 d^{5n-1}\tau^2\sigma + (\mathcal{C}_1 + \mathcal{C}_2)n\tau d^{n-1})$.

Proof. At each h-th step of algorithm, if there are more than one roots of the corresponding system in the positive orthant (the cost of estimating this is $\mathcal{C}_2$, we compute the corresponding partial quotients $l_h = (l_{h,1}, .., l_{h,n})$, where $\mathcal{L}(h_{h,i}) \leq \sigma_h$ (the cost of estimating this is $\mathcal{C}_1$ Then, for each polynomial of the system, f, we perform the shift operation $f(x_1 + l_1, \ldots, x_n + l_n)$, and then we split to 2^n subdomains. Let us estimate the cost of the last two operations.

A shift operation on a polynomial of degree $\leq d$, by a number of bitsize σ, increases the bitsize of the polynomial by an additive factor $nd\sigma$. At the h step of the algorithm, the polynomials of the corresponding system are of bitsize $\mathcal{O}(\tau + nd\sum_{i=1}^{h}\sigma_h)$, and we need to perform a shift operation to all the variables, with number of bitsize σ_{h+1}. The cost of this operation is $\widetilde{\mathcal{O}}_B(nd^n\tau + n^2d^{n+1}\sum_{k=1}^{h+1}\sigma_k)$, and since we have n polynomials the costs becomes $\widetilde{\mathcal{O}}_B(n^2d^n\tau + n^3d^{n+1}\sum_{k=1}^{h+1}\sigma_k)$, The resulting polynomial has bitsize $\mathcal{O}(\tau + nd\sum_{k=1}^{h+1}\sigma_k)$.

To compute the cost of splitting the domain, we proceed as follows. The cost is bounded by the cost of performing $n2^n$ operations $f(x_1+1, \ldots, x_n+1)$, which in turn is $\widetilde{\mathcal{O}}_B(nd^n\tau + n^2d^{n+1}\sum_{k=1}^{h+1}\sigma_k + n^2d^{n+1})$. So the total cost becomes $\widetilde{\mathcal{O}}_B(2^n n^2 d^n\tau + 2^n n^3 d^{n+1}\sum_{k=1}^{h+1}\sigma_k)$.

It remains to bound $\sum_{k=1}^{h+1}\sigma_k$. If σ is a bound on the bitsize of all the partial quotients that appear during and execution of the algorithm, then $\sum_{k=1}^{h+1}\sigma_k = \mathcal{O}(h\sigma)$.

[1]Actually the analysis holds even in the case where each l_i is the partial quotient of the positive imaginary part of a coordinate of a solution of the system.

Moreover, $h \le \#(T) = \mathcal{O}(n^2\tau d^{2n-1})$ (lem. 5.1), and so the cost of each step is $\widetilde{\mathcal{O}}_B(2^n n^5 d^{3n}\tau\sigma)$.

Finally, multiplying by the number of steps (lem. 5.1) we get a bound of $\widetilde{\mathcal{O}}_B(2^n n^7 d^{5n-1}\tau^2\sigma)$.

To derive the total complexity we have to take into account that at each step we compute some partial quotients and and we count the number of real root of the system in the positive orthant. Hence the total complexity of the algorithm is $\widetilde{\mathcal{O}}_B(2^n n^7 d^{5n-1}\tau^2\sigma + (\mathcal{C}_1 + \mathcal{C}_2)n\tau d^{n-1})$. $\square$

In the univariate case ($n = 1$), if we assume that (9) holds for real algebraic numbers, then the cost of $\mathcal{C}_1$ and $\mathcal{C}_2$ is dominated by that of the other steps, that is the splitting operations, and the (average) complexity becomes $\widetilde{\mathcal{O}}_B(d^3\tau)$ and matches the one derived in [18] (without scaling).

5.3 Further improvements

We can reduce the number of steps that the algorithm performs, and thus improve the total complexity bound of the algorithm, using the same trick as in [18]. The main idea is that the continued fraction expansion of a real root of a polynomial does not depend on the initial computed interval that contains all the roots. Thus, we spread away the roots by scaling the variables of the polynomials of the system by a carefully chosen value.

If we apply the map $(x_1, \dots, x_n) \mapsto (x_1/2^\ell, \dots, x_n/2^\ell)$, to the initial polynomials of the system, then the real roots are multiply by 2^ℓ, and thus their distance increase. The key observation is that the continued fraction expansion of the real roots does not depend on their integer part. Let ζ be the roots of the system, and γ, be the roots after the scaling. It holds $\gamma = 2^\ell \zeta$. From [8] it holds that

$$-\log \prod_{\zeta \in V} \Delta_i(\zeta) \le 2n\tau d^{2n-1}\lg(nd^{2n}) + 2nd^n \lg(nd^{2n}),$$

and thus

$$\begin{aligned}-\log \prod_{\zeta \in V} \Delta_i(\gamma) &= -\log 2^{R\ell} \prod_{\zeta \in V} \Delta_i(\zeta) \\ &\le (2n\tau d^{2n-1} + 2nd^n)\lg(nd^{2n}) - R\,\ell.\end{aligned}$$

If we choose $\ell = 2nd^{n-1}(d+\tau)\lg(nd^n)$ and assume that $R = d^n$ which is the worst case, then $-\log\prod_{\zeta\in V}\Delta_i(\gamma) = 0$. Thus, following the proof of Lem. 5.1, the number of steps that the algorithm is $\mathcal{O}(nd^n)$.

The bitsize of the scaled polynomials becomes $\widetilde{\mathcal{O}}(n^2 d^{n+1} + n^2 d^n \tau)$. The total complexity of algorithm is now

$$\widetilde{\mathcal{O}}_B(2^n n^5 d^{3n+1}\sigma + 2^n n^5 d^{3n}\tau + nd^n(\mathcal{C}_1 + \mathcal{C}_2)),$$

where σ the maximum bitsize of the partial quotient appear during the execution of the algorithm. If we assume that (9) holds for real algebraic numbers, then $\sigma = \mathcal{O}(1)$. Notice that in this case, when $n = 1$, the bound becomes $\widetilde{\mathcal{O}}_B(d^3\tau)$, which agrees with the one proved in [18].

The discussion above combined with Prop. 5.2 lead us to:

THEOREM 5.3. *The total complexity of the algorithm is* $\widetilde{\mathcal{O}}_B(2^n n^5 d^{3n+1}\sigma + 2^n n^5 d^{3n}\tau + nd^n(\mathcal{C}_1 + \mathcal{C}_2))$.

6. IMPLEMENTATION AND EXAMPLES

We have implemented the algorithm in the C++ library `realroot` of MATHEMAGIX[2], which is an open source effort that provides fundamental algebraic operations such as algebraic number manipulation tools, different types of univariate and multivariate polynomial root solvers, resultant and GCD computations, etc.

[2]`http://www.mathemagix.org/`

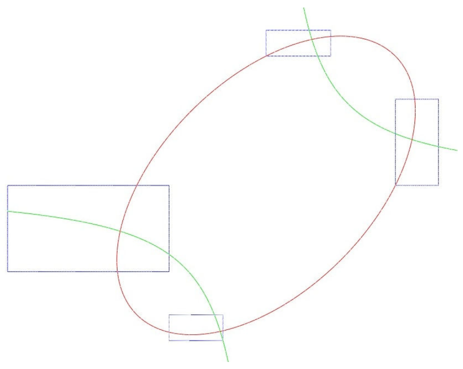

Figure 3: Isolating boxes of the real roots of (Σ_1).

The polynomials are internally represented as a vector of coefficients along with some additional data, such as a variable dictionary and the degree of the polynomial in every variable. This allows us to map the tensor of coefficients to the one-dimensional memory. The univariate solver that is used is the continued fraction solver; this is essentially the same algorithm with a different inclusion criterion, the Descartes rule. The same data structures is used to store the univariate polynomials, and the same shift/contraction routines. The univariate solver outputs the roots in increasing order, as a result of a breadth-first traverse of the subdivision tree. In fact, we only compute an isolation box for the smallest positive root of univariate polynomials and stop the solver as soon as the first root is found. Our code is templated and is efficiently used with GMP arithmetic, since long integers appear as the box size decreases.

The following four examples demonstrate the output of our implementation, which we visualize using AXEL[3].

First, we consider the system $f_1 = f_2 = 0$ (Σ_1), where $f_1 = x^2 + y^2 - xy - 1$, and $f_2 = 10xy - 4$. We are looking for the real solutions in the domain $I = [-2, 3] \times [-2, 2]$, which is mapped to $\mathbb{R}^2_+$, by an initial transformation. The isolating boxes of the real roots can be seen in Fig. 3.

In systems (Σ_2), (Σ_3), We multiply f_1 and f_2 by quadratic components, hence we obtain

$$(\Sigma_2)\begin{cases} f_1 = x^4 + 2x^2y^2 - 2x^2 + y^4 - 2y^2 - x^3y - xy^3 + xy + 1 \\ f_2 = 20x^3y - 10x^2y^2 - 10xy^3 - 8x^2 + 4xy + 4y^2 \end{cases}$$

and

$$(\Sigma_3)\begin{cases} f_1 = 10x^2y - 10xy^3 - 4x + 4y^2 \\ f_2 = x^4 - 2x^2y - 2x^2 + y^2x^2 - 2y^3 - y^2 - x^3y + \\ \quad +2xy^2 + xy + 2y + 1 \end{cases}$$

The isolating boxes of this system could be seen in Fig. 4. Notice, that size of the isolation boxes that are returned in this case is considerably smaller.

Consider the system (Σ_4), which consists of $f_1 = x^4 - 2x^2 - y^4 + 1$ and f_2, which is a polynomial of bidegree $(8, 8)$. The output of the algorithm, that is the isolating boxes of the real roots can be seen in Fig. 4.

[3]`http://axel.inria.fr`

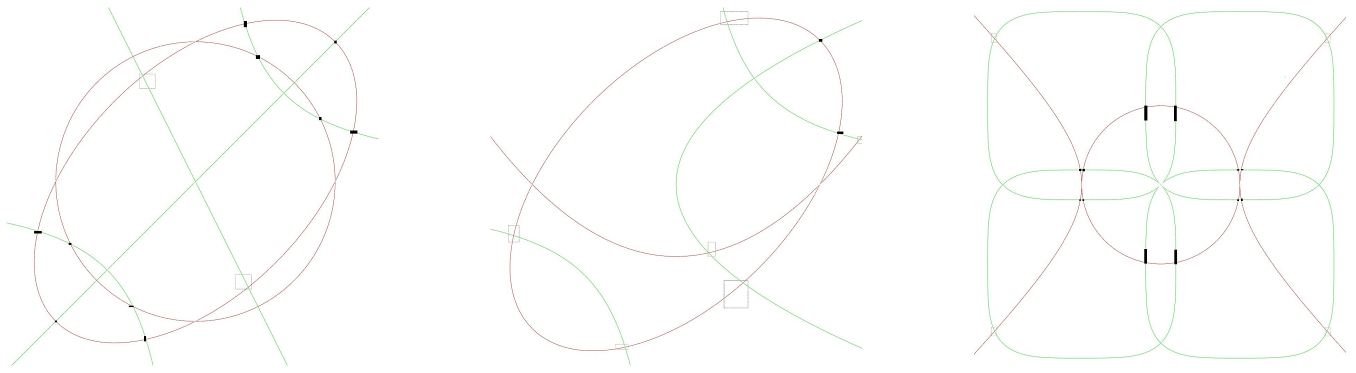

Figure 4: Isolating boxes of the real roots of the system Left: (Σ_2), Middle: (Σ_3), Right: (Σ_4).

System	Domain	Iters.	Subdivs.	Sols.	Excluded
Σ_1	$[0, 10]^2$	53	26	4	25
Σ_2	$[-2, 3]^2$	263	131	12	126
Σ_3	$[-2, 3]^2$	335	167	8	160
Σ_4	$[-3, 3]^2$	1097	548	16	533

Table 1: Execution data for Σ_1, Σ_2, Σ_3, Σ_4.

One important observation is the fact the isolating boxes *are not* squares, which verifies the adaptive nature of the proposed algorithm.

We provide execution details on these experiments in Table 1. Several optimizations can be applied to our code, but the results already indicate that our approach competes well with the Bernstein case.

Acknowledgements
The first and second author were supported by Marie-Curie Initial Training Network SAGA, [FP7/2007-2013], grant [PITN-GA-2008-214584]. The third author was supported by contract [ANR-06-BLAN-0074] "Decotes".

7. REFERENCES

[1] A. Alesina and M. Galuzzi. New proof of Vincent's thm. *Enseignement Mathématique*, 44:219–256, 1998.

[2] M. Bartoň and B. Jüttler. Computing roots of polynomials by quadratic clipping. *Comp. Aided Geom. Design*, 24:125–141, 2007.

[3] E. Bombieri and A. van der Poorten. Continued fractions of algebraic numbers. In *Computational algebra and number theory (Sydney, 1992)*, pp. 137–152. Kluwer Acad. Publ., Dordrecht, 1995.

[4] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the Descartes method. In *ISSAC 2006*, pp. 71–78. ACM, New York, 2006.

[5] G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *Proc. of 6th ACM Symposium on Solid Modelling and Applications*, pp. 1–10. ACM Press, 2001.

[6] I. Emiris, M. Hemmer, M. Karavelas, S. Limbach, B. Mourrain, E. P. Tsigaridas, and Z. Zafeirakopoulos. Cross-benchmarks of univariate algebraic kernels. ACS-TR-363602-02, INRIA, MPI and NUA, 2008.

[7] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, ed., *Reliable Implementations of Real Number Algorithms: Theory and Practice, LNCS* vol. 5045, pp. 57–82. Springer Verlag, 2008.

[8] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. The dmm bound: multivariate (aggregate) separation bounds. Technical report, INRIA, March 2009.

[9] J. Garloff and A. P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Journal of Nonlinear Analysis*, 47(1):167–178, 2001.

[10] A. Khintchine. *Continued Fractions.* University of Chicago Press, Chicago, 1964.

[11] M. Marden. *Geometry of Polynomials.* American Mathematical Society, Providence, RI, 1966.

[12] B. Mourrain and J. Pavone. Subdivision methods for solving polynomial equations. Special issue in honor of Daniel Lazard. *JSC* 44(3):292 – 306, 2009.

[13] B. Mourrain, F. Rouillier, and M.-F. Roy. *Bernstein's basis and real root isolation*, pp. 459–478. MSRI Publications. Cambridge University Press, 2005.

[14] V. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.

[15] V. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *JSC*, 33(5):701–733, 2002.

[16] V. Sharma. Complexity of real root isolation using continued fractions. *Theor. Comput. Sci.*, 409(2):292–310, 2008.

[17] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Comput. Aided Geom. Design*, 10(5):379–405, 1993.

[18] E. P. Tsigaridas and I. Z. Emiris. On the complexity of real root isolation using Continued Fractions. *Theoretical Computer Science*, 392:158–173, 2008.

[19] A. van der Poorten. An introduction to continued fractions. In *Diophantine analysis*, pp. 99–138. Cambridge University Press, 1986.

[20] J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *Proc. Annual ACM ISSAC*, pp. 40–47, 1997.

[21] M. N. Vrahatis. A short proof and a generalization of Miranda's existence theorem. *Proceedings of the American Mathematical Society*, 107(3):701–703, 1989.

[22] C. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, New York, 2000.